Introducing Development with Hive

By the end of this video, you should understand the essentials about Hive after exploring its architecture.
We will also spend some time going over HiveQL, and compare Hive to traditional relational database management systems.
Apache Hive is an open-source data warehouse system for querying and
analyzing large data sets stored in Hadoop files.
Hadoop is a framework for handling large data sets
in a distributed computing environment.
Apache Hive allows you to query and manage large data sets in distributed storage by using a SQL-like language called HiveQL.
Hive, like Pig, is an obstruction on top.
HiveQL looks a lot like ANSI SQL.
That means if you know Transact-SQL,
you'll feel comfortable learning Hive.
Because the results look like a standard relational database result set, various vendors have created
ODBC drivers that interact with Hive results.
With Hive, you can execute statements using either the Hive command line interface on the Hadoop cluster or
via an interactive web console like the Hive Interactive on HDInsight.
Applications can send queries and
return results via either ODBC or JDBC drivers.
Internally, the HIVE compile, optimizer, an executor,
translate HiveQL statements into a directed graph of MapReduce jobs that are submitted to the Hadoop cluster's headnote for execution.
As part of the optimization phase,
developers can create customized mappers and
reducers to extend the functionality of HiveQL.
This figure shows the major components of Hive and
its interactions with Hadoop, which include the UI,
which is the user interface for users to submit queries and
other operations to the system.
Also the driver,
which is the component which receives the queries.
This component implements the notion of session handles, and provides execute and fetch APIs modeled on JDBC/ODBC interfaces.
The compiler is the component that parses the query,
the semantic analysis on different query blocks and
query expressions, and
then eventually generates an execution plan with the help of
the table and partition metadata looked up from the Metastore.

The Metastore is the component that stores all of the structure information of the various tables and
partitions in the warehouse, including column and
column type information, the serializers and
deserializers necessary to read and write data, and
the corresponding HDFS files where the data is stored.
Then there's the execution engine,
which executes the execution plan created by the compiler.
The plan is a DAG of stages.
The execution engine manages the dependencies between these different stages of the plan, and
executes these stages on the appropriate system components.
On a very high level,
Hive is just creating a MapReduce plan for the query and
then submitting it to the job tracker.
A Hive query can be submitted through many channels, Hue, CLI, Thrift, the driver is the core which compiles the input.
It gets the table definitions from the Metastore,
then the driver optimizes the computation required, then it
creates MapReduce jobs and submits them to the job tracker.
Data and Hive is organized into tables.
These are analogous to tables and relational databases.
Tables can be filtered, projected, joined in union.
Additionally, all the data of a table is stored in the directory
in HDFS.
Hive also supports the notion of external tables,
wherein a table can be created on preexisting files or
directories in HDFS by providing the appropriate location to
the table creation DDL.
Then you have partitions, each table can have one or more
partition keys, which determine how the data is stored.
For example, a table T with a date partition column DS,
has files with data for a particular date stored in
the table location/ds equals date directory in HDFS.
Next is buckets, data in each partition may in turn be divided
into buckets based on the hash of a column in the table.
Each bucket is stored as a file in the partition directory.
Bucketing allows the system to efficiently evaluate queries
that depend on a sample of data,
these are queries that use the sample clause in the table.
This sample file includes customer record data in Hive.
And this slide shows the sample data of the order record file.
The standard workflow for using Hive is as follows.
First, you create a table on top of your data.
If you want your data to be preserved at the location of
the source table, you can create an external table.

If you include the external clause, the data for
the table is moved into the Hive data warehouse.
Data can be imported and
exported into the table with import and
export commands, you can also use the load data inpath command
to associate a table with data in your Hadoop cluster.
Finally, Hive offers a rich set of select command features for
querying data as the order by, group by, join,
union, and subquery clauses.
For guidance on syntax and
usage, refer to the Hive language manual on apache.org.
In Hive external tables, data remains outside the Hadoop
cluster, and T-SQL skills can be used to query the Hive data.
There are a few things that you need to know about the Hive
internal table and external table.
The internal table manages the underlying data files for
the table, and
physically deletes the data files when you drop the table.
If you specify the external clause,
it treats the table as an external table,
where the data files are typically produced outside it
and queried from their original locations in HDFS, and
it leaves the data files in placed when you drop the table.
Typically, for an external table you include a location clause
to specify the path to the HDFS directory where it reads and
writes the files for the table.
You can create multiple views of the same table,
multiple tables can point to the same data.
Load operations are currently pure copy, and
move operations that move data files into
locations corresponding to Hive tables.
You can use local inpath or
source destination to load local data.
If the key word local is specified, then the load
command will look for file path in the local file system.
If a relative path is specified, however, it will be interpreted
relative to the user's current working directory.
You can also load data from other Hive Tables.
Query results can be inserted into tables by using
the insert clause.
The code listed toward the bottom of this slide,
load data local inpath, then there's the path,
home/dataset/customers_orders_h-
ive,
overwrite into table customer partition year=2015.
Note that all the data has been loaded into the year equals 2515

partition.
Now the year can be used in queries for
partition elimination as if it was a regular table column.
This slide shows the select query to drill down into
the customer details in the Hive table.
Select is the projection operator in SQL.
Hive provides an SQL-like language called HiveQL, with
schema on read and transparently converts queries to MapReduce.
HiveQL offers extensions not in SQL,
including multi-table inserts and create table as select.
But it only offers basic support for
SQL like Syntax, with all and distinct query statements,
it also includes support for partitioning and indexes.
HiveQL lacks support for transactions and
materialize views, and only has limited subquery support.
It does have support for group by, sort, order,
cluster, distribute by, JOIN, with Hive JOINs,
JOIN optimization, outer JOIN behavior.
Plus union table sample, subqueries,
virtual columns, operators and UDFs, lateral view windowing,
over, and analytics plus common table expressions.
Here we see an example select query in HiveQL,
selecting the name and billing city of customers with shipping
addresses in Texas or Illinois and group by city.
You can see the results of the query in the adjacent screen.
This example, select query in HiveQL, finds names of products
purchased by costumers who's shipping address is in Texas.
You can see the results of this query in the adjacent screen.
HDInsight can run HiveQL jobs using a variety of methods.
Use the following table to decide which method is right for
you, then follow the link for a walk-through.
Let's discuss the differences between Hive and
RDBMS, which is a traditional relational database.
Examples of traditional relational databases are MySQL,
PostgreSQL, Oracle 11g, SQL server, etc.
Below are the key features of Hive that defer from RDBMS.
Hive resembles a traditional database by supporting
a SQL interface, but it is not a full database.
Hive can be better called a data warehouse instead of a database.
Hive enforces the schema on read time, whereas,
RDBMS enforces the schema on write time.
In RDBMS, a table schema is enforced at data load time.
If the data being loaded doesn't conform to the schema,
then it is rejected, this design is called schema on write.
But Hive doesn't verify the data when it is loaded, but
rather when it is retrieved, this is called schema on read.

You can access the RDBMS DW through syntax
with a language like SQL, whereas in Hive,
you can use HiveQL, both support indexes and partitions.
Referential integrity and
data optimization is not supported in Hive.
Hive is based on the notion of write once, read many times.
But RDBMS is designed for read and write many times.
In RDBMS, record level updates, insertions and
deletes, transactions and indexes are possible.
Whereas, these are not allowed in Hive because Hive was built
to operate over HDFS data using MapReduce, or
full table scans are the norm and a table update is achieved
by transforming the data into a new table.
As Hadoop is a batch oriented system,
Hive doesn't support OLTP or online transaction processing,
but it is closer to OLAP, which is online analytical processing.
But not ideal since there is a significant latency
between issuing a query and
receiving a reply due to the overhead of MapReduce jobs, and
due to the size of the data sets Hadoop was designed to serve.
And RDBMS is best suited for dynamic data analysis, and
where fast responses are expected, but Hive is suited for
data warehouse applications where relatively static
data is analyzed, fast response times are not required, and
when the data is not changing rapidly.
This table and diagram show the recent Hive releases,
along with the improvements they brought with them.
It is possible to author, submit, and debug Hive
queries to an HDInsight cluster using the HDInsight Tools for
Visual Studio.
To do this, at a minimum you need Visual Studio 2013 with
either the community, professional, premium, or
ultimate editions, or
have Visual Studio 2015 Community or Enterprise.
The HDInsight Tools for Visual Studio allows you to connect
to your HDInsight clusters, perform some basic management
operations, and run Hive queries.
From Server Explorer, you can see the default storage account
in any linked storage accounts.
If you expand the default storage account,
you can see the containers underneath it.
HDInsight Tools for
Visual Studio is capable of validating a Hive script locally
without having to connect to a live cluster.
HDInsight Tools for Visual Studio also enables users
to see what's inside the Hive job by collecting and

surfacing the yarn logs of certain Hive jobs.
You can use the to create a Hive table or use Hive queries.
There are two ways to create and run Hive queries,
create ad hoc queries and create Hive application.
You can view job queries, job outputs, job logs, and
yarn logs for Hive jobs.